State of the Art on: Quantum Computing and hardness of computation of combinatorial functions

Marco Venere, marco.venere@mail.polimi.it

1. Introduction to the research topic

The main areas related to the presented research topic are Electronic Design Automation (EDA) and Quantum computing.
For what concerns EDA, it is a research field dedicated to the development of algorithms and software tools for the automation of tasks related to the design of digital electronic circuits. In particular, the *frontend* of such instruments is the logic design flow which, given a high-level description of the circuit, produces a *netlist*, which is a document defining all the instances of the used blocks and their interconnections. Each of these blocks represents a specific Boolean function. The EDA *backend* is instead the physical design flow which, given a netlist, defines the final schematic of the circuits, i.e., establishing which library components to use, how to place them in the chip's core area (Placement) and how to interconnect them (Routing). The available combinatorial components of the libraries implement elementary Boolean functions, therefore each logic block of the netlist must be mapped to one or more library components (Technology Mapping), in such a way as to obtain an equivalent final circuit.
In order to reduce the amount of used resources, as well as the synthesis time and the development costs, some optimization tasks are performed before producing the final gate description of the product. Among them, the Boolean Matching Problem, in the Technology Mapping, regards the comparison of the Boolean function to synthesize and the given library gates, in order to find equivalence under input negation and permutation and output negation (NPN-equivalence). The most efficient algorithms dedicated to the solution of this problem are NP-hard (belonging to $O(2^{n+1}n!)$ in the size of the input of the function), thus being excessively time-consuming, and can only be performed on a limited amount of input bits.
For what concerns Quantum Computing, it is a model of computation whose operations can harness the phenomena of quantum mechanics, such as superposition, interference, and entanglement. Quantum algorithms have often proven to offer a noticeable speedup w.r.t. their analogue classical implementations, and are becoming an essential tool to tackle significant problems.
The main journals describing these research topics, and their 2-year impact factor metric, according to which they have been chosen, are:

1. ACM Transactions on Quantum Computing, which publishes high-impact, original research papers and select surveys on topics in quantum computing and quantum information science. Its 2-year Impact Factor is 2.8 [1], therefore entering the top 6% of international journals.
2. IEEE Transactions on Quantum Engineering, which publishes regular, review, and tutorial articles based on the engineering applications of quantum phenomena, including quantum computation, information, and software. Its 2-year Impact Factor is 2.8 [3], therefore entering the top 6% of international journals.
3. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, which publishes papers of interest to individuals in the area of computer-aided design of integrated circuits and systems. Its 2-year Impact Factor is 2.2 [2], therefore entering the top 8% of international journals.

For what instead concerns the main conferences, they are considered based on the Hirsch index (H-index), which is defined as the maximum value of h such that the given conference has published at least h papers that have each been cited at least h times. The top conferences that have been identified are:

1. Design Automation Conference (DAC), which is an annual event, a combination of a technical conference and a trade show, both specializing in electronic design automation. DAC is the oldest and largest conference in EDA, started in 1964. Its H-index is 122 [4].
2. Design, Automation & Test in Europe (DATE), which is a yearly conference on the topic of electronic design automation. It is typically held in March or April of each year, alternating between France and Germany. Its H-index is 92 [5].
3. Quantum Information Processing (QIP), which is an annual conference about quantum computation and information, usually held in January. Its goal is to represent the preceding year's best research in the area, in the form of both invited and contributed talks. Its H-index is 60 [8].
4. Asia and South Pacific Design Automation Conference (ASP-DAC), which is a yearly conference on the topic of electronic design automation. It is typically held in late January in the Far East, as the name implies. Its H-index is 58 [6].

1.1. Preliminaries

In order to better understand the context, it is necessary to define the Boolean Matching Problem [12].
Boolean Matching Problem. Let g be the Boolean function computed by some element of a technology library. Let f be another Boolean function, called the target function. Boolean matching is the problem of determining whether g can be used to implement f. We assume that f and g are represented by Binary Decision Diagrams (BDDs). Clearly, g can implement f if $g \approx f$, i.e., g and f are *equivalent*. In addition to the strict coincidence between g and f, there are weaker notions of equivalence that are more useful for the Boolean matching context, and in particular:
NPN-equivalence between two Boolean functions. Given two Boolean functions, f and g, with the same number of inputs, then f and g are said to be NPN-equivalent iff f can be formed from g by negating zero or more inputs, permuting inputs or negating the output, or by combinations of these transformations.
Another concept of great importance is the *canonical form* of a Boolean function, which is a Boolean function describing the NPN-equivalence class of Boolean functions to which the given function belongs.
Additionally, the *signature* of a Boolean function is a mathematical object representing a property of the given function [16].
Finally, the *Walsh-Hadamard Transform* is an example of a generalized class of Fourier transforms, that can be applied to Boolean functions [13].
Implementation and technological tools
For what concerns the implementation and technological tools used in the field, to design quantum circuits and simulate or run them on real quantum hardware, frameworks can be used like:

1. Qiskit [7], which is an open-source SDK for working with quantum computers at the level of pulses, circuits, and algorithms. It provides tools for creating and manipulating quantum programs and running them on prototype quantum devices on IBM Quantum Experience or on simulators on a local computer.
2. QX Simulator [9], which is a universal quantum computer simulator developed at QuTech by Nader Khammassi. The QX allows quantum algorithm designers to simulate the execution of their quantum circuits on a quantum computer. The simulator defines a low-level quantum assembly language.

1.2. Research topic

The research topic is the quantum acceleration of the solver for the previously described Boolean Matching Problem (BMP). The objective is therefore to design a quantum circuit that effectively solves this kind of tasks while taking advantage of quantum principles, such as superposition.
Indeed, the BMP is a task of great importance in EDA software suites, since it allows to reduce development costs by reusing preexisting logic components. Yet, it is computationally expensive, thus requiring further studies on possible ways to accelerate solvers' algorithms.

2. Main related works

2.1. Classification of the main related works

The main works related to the resolution of the Boolean Matching Problem, by using a classical approach, can be split into 3 principal branches, which exploit specific properties of the given Boolean Functions, in order to reduce the size of the search space in which to find NPN-equivalent functions:

1. Reduction to the canonical form: Boolean Functions can be associated with their canonical form, and it has been proven that functions belonging to the same equivalence class are associated with the same canonical form. Therefore, computing such a form for the target Boolean Function and comparing it against the canonical forms of the functions belonging to the given libraries is theoretically proven to solve this task. Several ways have been implemented to output a correct canonical form. State-of-the-Art (SoA) solutions currently consider the Binary Decision Diagram (BDD) as a valid representation of the function, in order to improve performance.
2. Signature Pruning: such a method performs a search in the space of the potentially NPN-equivalent Boolean Functions after reducing it by pruning: indeed, there exist specific signatures, i.e. alternative representations for functions, which can be proven to be invariant for NPN-transformations. Therefore, pruning can be done on the search tree, based on comparison of signatures.
3. Spectral-based methods: they exploit the property that NPN-equivalence of Boolean Functions translates into equivalence under coefficient permutation into the sequence domain of the Walsh Transform. Therefore, an exploration of the permutation tree of the Walsh coefficients is performed.

It is also important to notice that hybrid approaches can be provided, and are present in several works.

For what concerns the first branch of methods, useful works in the bibliography are [12] and [11], whilst for the second branch useful works are [10], [16] and [15]. Finally, for the third branch, [13] is presented. Some of these papers, even though attributed to a specific branch, may even be considered, for certain aspects, as hybrid. For what instead concerns the aspects related to the quantum acceleration of EDA algorithms, a noticeable work regards the implementation of a quantum circuit addressing the Max-Cut Problem [14]. Yet, no relevant materials have been found related to the Boolean Matching Problem.

2.2. Brief description of the main related works

Efficient Boolean Function Matching
This paper [12] proposes an algorithm for Boolean matching in which functions are translated to their canonical form and then compared. The algorithm to obtain a canonical form is polynomial in the size of the Binary Decision Diagram (BDD). Their implementation on FPGAs has obtained a good speedup w.r.t. other existing frameworks. It starts by considering phase matching (i.e., only negation of input), and it notices that, if the canonical forms of each component of the library are computed, then it is possible to see if a given function phase-matches a component of the library in sublinear time (e.g., search in a hash table of canonical forms of the library components). It also considers permutation equivalence.
A Transform-Parametric Approach to Boolean Matching
This paper [11] focuses on P-equivalence Boolean matching. In particular, it produces a framework that considers both spectral-based and canonical-form-based approaches to verify matches among Boolean functions.
For what concerns spectral-based methods, they are based on the fact that P-equivalence (and, in general, NPN-equivalence) in the time domain translates to equivalence under coefficient permutations in the sequence domain of the Walsh Transform. Yet, algorithms based on this peculiarity have exponential time complexity.
For what concerns the canonical-form-based methods, they define classes of equivalence of Boolean functions under permutation, and consider the lexicographical maximum of the class as the canonical form of the class itself. After defining some utility functions, the authors present an algorithm to get a canonizing permutation.

Since, in order to compute all the possible permutations in a class and find the maximal one, an algorithm of complexity $O(n!)$ is necessary, another algorithm, which uses a bijective linear transformation of the functions, is used. In particular, the *cofactor transformation* is considered, which gives an efficient way to compute the canonical form.
They finally describe a generalized algorithm for the computation of the canonizing permutation, which produces a list of candidate canonizing permutations. From these, the lexicographical maximum is selected.
In addition to the cofactor transformation, other possible transformations have been considered, e.g., the usage of the Walsh coefficients. The Walsh Transform is defined by means of the Hadamard Matrix. The cofactor representation uses instead a C_n matrix, which appears to be very similar to the Hadamard Matrix.
Subsequently, the exploration of the permutation tree is described, by introducing specific cost functions, like a canonical-based one, associated with the onset representation of the function, and based on the minimization of the integer number corresponding to the natural binary encoding of the string obtained by juxtaposing the minterms resulting from a permutation of the input variables.
Another transformation is the *shifted cofactor representation*, which has been proven, in the testing phase, to outperform the cofactor representation.
Signature based Boolean matching in the presence of don't cares
This paper [10] adds the case of incompletely specified functions, making use of *don't cares*. It describes a signature-based approach in which the satisfy count is used to generate signatures for variables (first order) and also for pair of variables of the same functions (second order). While for completely specified functions the satisfy count is an integer, for incompletely specified functions the satisfy count can vary within a range of numbers. Two ISFs can be matched only if their corresponding first signature ranges have a non-empty intersection. This fact, in addition to the usage of the second signatures, is used to prune the search space. BDDs are used to represent the functions and extract signature information. In particular, a recursive relation is employed to compute signatures. Therefore, the phase of the functions is determined, the input match graph is created and a matching algorithm is applied, which reduces the graph and performs branch and bound.
An efficient NPN Boolean matching algorithm based on structural signature and Shannon expansion This paper [16] presents a signature-based Boolean matching algorithm that takes advantage of the structural signature (SS) vector, which comprises a first-order signature value, two symmetry marks, and a group mark.
After considering that two equivalent functions need to have the same first-order signature, it defines an algorithm that uses SS vectors and Shannon decomposition to search for variable mappings and to form possible transformations. In particular, it computes the phase of the variables x_i: if $|f_{x_i}| > |f_{\overline{x_i}}|$, the phase of x_i will be positive and defined as $+1$. If, instead, $|f_{x_i}| < |f_{\overline{x_i}}|$, it will be negative and defined as -1. Otherwise, it will be considered undetermined and referred to as 0. Therefore, the algorithm will create a variable mapping between x_i of Boolean function f and x_j of Boolean function g if these two variables have the same SS value. To reduce the search space, we should determine the phases of as many variables as possible. If the first signature value of some variable can be changed, then the phases may be determined and the problem is solved.
If the variable mapping set of x_i of f is a multiple-mapping set, then multiple variables of g with the same SS value as that of x_i exist, and we must attempt multiple variable mappings for x_i. Since undetermined phases cause the search space to grow, we utilize identified variables to update the SS vector and cause unidentified variables to have different SS values. Thus, updating the SS vector is an important step.
The algorithms apply Shannon decompositions, which do not alter the NP-equivalence between functions, to form *cubes* described by the splitting variables of the decompositions. These variables are used to update the SS vectors for phase assignment and search for variable mappings.
Since each transformation between functions has n variable mappings, the comparison of the SS values for each pair of variables is performed to check whether they have the same SS value, number of variables in the symmetry class, and group number. In that case, they have a mapping.
Therefore, the number of possible combinations to consider is very high. It can be reduced by finding incorrect transformations. This can be achieved by comparing SS vectors (if they are not the same, the transformation can be discarded), and phase collision is checked too.

Experimental results show that an effective speedup has been obtained in the matching speed, with respect to other signature-based state-of-the-art algorithms. In particular, good scalability has been proven, in terms of both space and time complexity. As possible next steps, authors suggest support for *don't cares* and multiple-output Boolean function matching.
A New Pairwise NPN Boolean Matching Algorithm Based on Structural Difference Signature This paper [15] considers the usage of the *structural difference signature* (SDS) of a Boolean function as a tool to reduce the search space in the Boolean matching process. Based on [16], it reimplements the structural signature (SS) vector, by introducing the concept of Boolean difference, and the consequent Boolean difference signature. This signature, like the cofactor signature, can be used to distinguish variables.
After this, it defines the independence property of variables, observing that NP transformations of variables preserve their independence. It, therefore, defines the structural difference signature vector. It consists of the SS vector plus a Boolean difference mark. It then considers that two NPN-equivalent Boolean functions have the same SDS vectors, and, if two variables x_i and x_j have the same SDS values, then there exists a possible variable mapping between them.
By taking advantage of such definitions and properties, an SDS-based Boolean matching algorithm is described, in which variable mappings for each variable that has not been identified are considered. Once a variable has been identified (i.e., its phase and variable mapping are determined in a transformation), the next variable is analyzed. During the search, symmetry properties of variables are exploited to prune the search space. In particular, a search tree is navigated and pruned whenever possible.
Experimental results show the effectiveness of the algorithm and the overall speedup achieved w.r.t. the state of the art.
Boolean Function Matching using Walsh Spectral Decision Diagrams This paper [13] considers two approaches for Boolean matching. They are the usage of Luks' hypergraph method and the Walsh Spectral Decision Diagram (SDD). The latter is described and implemented by the authors, who show its speedup w.r.t. the former solution. In particular, after defining the Walsh Spectrum, an overview of the Luks' Method is given, which is based on hypergraph isomorphism checking. Then, a formalism related to the Walsh coefficients is presented. The results of the Walsh spectrum method are finally compared to Luks' method.
Max-Cut Problem Implementation and Analysis on a Quantum Computer This paper [14] presents an approach to the quantum acceleration of the Max-Cut Problem, which is a combinatorial optimization problem consisting in partitioning two sets of graph nodes such that the number of edges between the sets is maximum. The Quantum Approximate Optimization Algorithm (QAOA) has been used to achieve the final quantum circuit, since it is a quantum algorithm specialized in solving combinatorial problems. Testing for this algorithm has been executed on real IBM quantum computers, as well as IBM quantum simulators, and a noise analysis due to qubits' physical properties has been performed too. The transpilation of the quantum circuits and their mapping have been shown. Among the limits found in the current implementation, there is the effect of noise on the results provided by the circuit, even though some noise mitigation techniques have been applied.

2.3. Discussion

The main open issues regarding the implementation of a quantum accelerated version of BMP are related to the specific quantum properties to use, since the involvement of quantum parallelism and superposition may require the adoption of a new approach toward the solution of this optimization task.
Concurrently, there are issues related to the limits of current quantum hardware. Such issues have also been shown by [14], and are due to the scarce amount of available qubits, as well as their physical noise. Functional testing is not strictly affected and can still be performed, but the size of the input onto which testing is possible may change based on available quantum hardware.

References

[1] Acm transactions on quantum computing impact factor. `https://exaly.com/journal/68559/acm-transactions-on-quantum-computing/`. Accessed: 2022-11-29.

[2] Ieee transactions on computer-aided design of integrated circuits and systems. `https://exaly.com/journal/14323/ieee-transactions-on-computer-aided-design-of-integrated-circuits-and-systems`. Accessed: 2022-11-29.

[3] Ieee transactions on quantum engineering impact factor. `https://exaly.com/journal/46055/ieee-transactions-on-quantum-engineering/where-cited`. Accessed: 2022-11-29.

[4] Proceedings - design automation conference. `https://www.scimagojr.com/journalsearch.php?q=110497&tip=sid&clean=0`. Accessed: 2022-11-29.

[5] Proceedings -design, automation and test in europe, date. `https://www.scimagojr.com/journalsearch.php?q=5400152603&tip=sid&clean=0`. Accessed: 2022-11-29.

[6] Proceedings of the asia and south pacific design automation conference, asp-dac. `https://www.scimagojr.com/journalsearch.php?q=77904&tip=sid&clean=0`. Accessed: 2022-11-29.

[7] Qiskit. `https://qiskit.org/`. Accessed: 2022-11-29.

[8] Quantum information processing- impact score, overall ranking, h-index, sjr, rating, publisher, issn, and other important metrics. `https://www.resurchify.com/impact/details/24831`. Accessed: 2022-11-30.

[9] The qx simulator. `http://quantum-studio.net/`. Accessed: 2022-11-29.

[10] Abdollahi, A. Signature based boolean matching in the presence of don't cares. In *2008 45th ACM/IEEE Design Automation Conference* (2008), pp. 642–647.

[11] Agosta, G., Bruschi, F., Pelosi, G., and Sciuto, D. A transform-parametric approach to boolean matching. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28*, 6 (2009), 805–817.

[12] Burch, and Long. Efficient boolean function matching. In *1992 IEEE/ACM International Conference on Computer-Aided Design* (1992), pp. 408–411.

[13] Moore, J., Fazel, K., Thornton, M. A., and Miller, D. M. Boolean function matching using walsh spectral decision diagrams. In *2006 IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software* (2006), pp. 127–130.

[14] Verghese, A., Byron, D., Amann, A., and Popovici, E. Max-cut problem implementation and analysis on a quantum computer. In *2022 33rd Irish Signals and Systems Conference (ISSC)* (2022), IEEE, pp. 1–6.

[15] Zhang, J., Yang, G., Hung, W., Wu, J., and Zhu, Y. A new pairwise npn boolean matching algorithm based on structural difference signature. *Symmetry 11* (12 2018), 27.

[16] Zhang, J., Yang, G., Hung, W. N., Zhang, Y., and Wu, J. An efficient npn boolean matching algorithm based on structural signature and shannon expansion. *Cluster Computing 22*, 3 (2019), 7491–7506.